K-Nearest Neighbors

Nipun Batra

July 20, 2025

IIT Gandhinagar

null

null

Parametric vs Non-Parametric Models

	Parametric	Non-Parametric
Parameter	Number of parameters is fixed w.r.t dataset size	Number of parameters grows w.r.t. to an increase in dataset size
Speed	Quicker (as the number of parameters are less)	Longer (as number of parameters are less)
Assumptions	Strong Assumptions (like linearity in Linear Regression)	Very few (sometimes no) assumptions
Examples	Linear Regression	KNN, Decision Tree

Lazy vs Eager Strategies

	Lazy	Eager
Train Time	0	$\neq 0$
Test	Long (due to comparison with train data)	Quick (as only “parameters” are involved)
Memory	Store/Memorise entire data	Store only learnt parameters
Utility	Useful for online settings	
Examples	KNN	Linear Regression, Decision Tree

Important Considerations

- What are the **features** that will be considered for data similarity?

Important Considerations

- What are the **features** that will be considered for data similarity?
- What is the **distance metric** that will be used to calculate data similarity?

Important Considerations

- What are the **features** that will be considered for data similarity?
- What is the **distance metric** that will be used to calculate data similarity?
- What is the **aggregation function** that is going to be used?

Important Considerations

- What are the **features** that will be considered for data similarity?
- What is the **distance metric** that will be used to calculate data similarity?
- What is the **aggregation function** that is going to be used?
- What are the **number of neighbors** that you are going to take into consideration?

Important Considerations

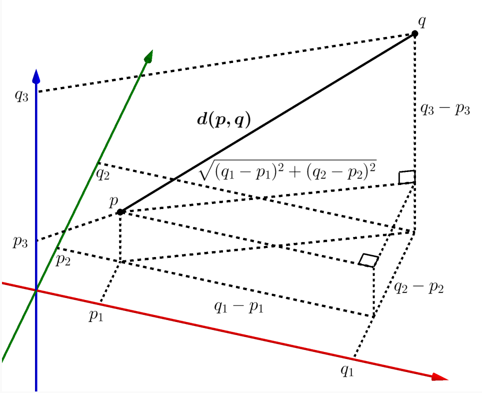
- What are the **features** that will be considered for data similarity?
- What is the **distance metric** that will be used to calculate data similarity?
- What is the **aggregation function** that is going to be used?
- What are the **number of neighbors** that you are going to take into consideration?
- What is the **computational complexity** of the algorithm that you are implementing?

Important Considerations: Distance Metric

The Distance Metric acts as a *measure of similarity* between the points.

Important Considerations: Distance Metric

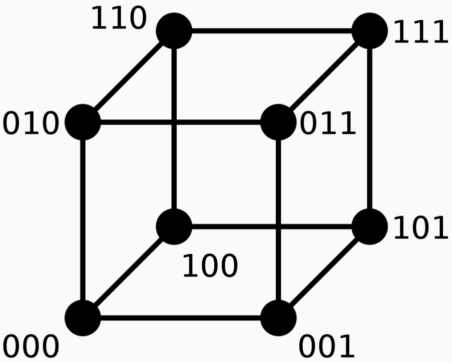
The Distance Metric acts as a *measure of similarity* between the points.



Euclidean Distance

Important Considerations: Distance Metric

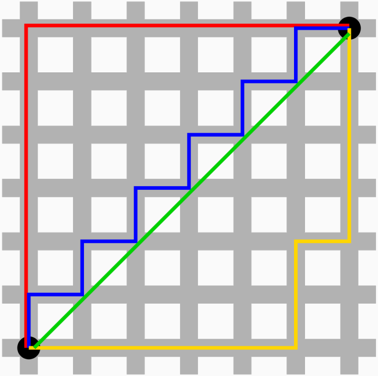
The Distance Metric acts as a *measure of similarity* between the points.



Hamming Distance

Important Considerations: Distance Metric

The Distance Metric acts as a *measure of similarity* between the points.



Manhattan Distance

Important Considerations: Value of K

Choosing the correct value of K is difficult.

Important Considerations: Value of K

Choosing the correct value of K is difficult.

Low values of K will result in each point having a very high influence on the final output $\implies$ noise will influence the result

Important Considerations: Value of K

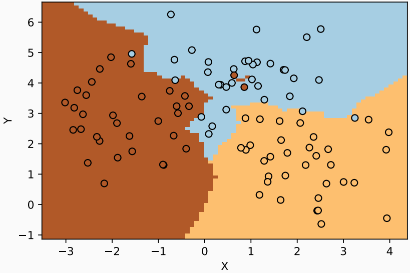
Choosing the correct value of K is difficult.

Low values of K will result in each point having a very high influence on the final output $\implies$ noise will influence the result

High values of K will result in smoother decision boundaries $\implies$ lower variance but also higher bias

Important Considerations: Value of K

Important Considerations: Value of K



K = 3

Important Considerations: Value of K

Aggregating data

There are different ways to go about aggregating the data from the K nearest neighbors.

- Median

Aggregating data

There are different ways to go about aggregating the data from the K nearest neighbors.

- Median
- Mean

Aggregating data

There are different ways to go about aggregating the data from the K nearest neighbors.

- Median
- Mean
- Mode

KNN Algorithm

- Keep the entire dataset: (x, y)

KNN Algorithm

- Keep the entire dataset: (x, y)
- For a query vector q:

KNN Algorithm

- Keep the entire dataset: (x, y)
- For a query vector q:
    1. Find the k-closest data point(s) x^*

KNN Algorithm

- Keep the entire dataset: (x, y)
- For a query vector q:
  1. Find the k-closest data point(s) x^*
  2. Predict y^*

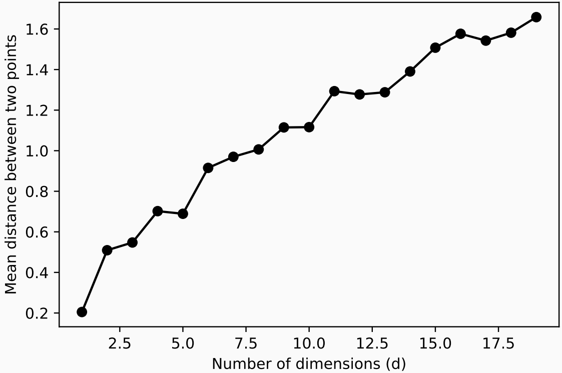
Curse of Dimensionality

With an increase in the number of dimensions:

Curse of Dimensionality

With an increase in the number of dimensions:

1. the distance between points starts to increase



For a unifromly random dataset

Curse of Dimensionality

With an increase in the number of dimensions:

1. the distance between points starts to increase

Curse of Dimensionality

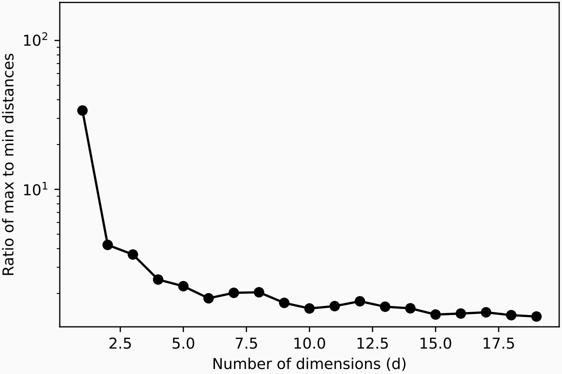
With an increase in the number of dimensions:

1. the distance between points starts to increase
2. the variation in distances between points starts to decrease

Curse of Dimensionality

With an increase in the number of dimensions:

1. the distance between points starts to increase
2. the variation in distances between points starts to decrease



For a unifromly random dataset

Curse of Dimensionality

With an increase in the number of dimensions:

1. the distance between points starts to increase

Due to this, distance metrics lose their efficacy as a similarity metric.

Curse of Dimensionality

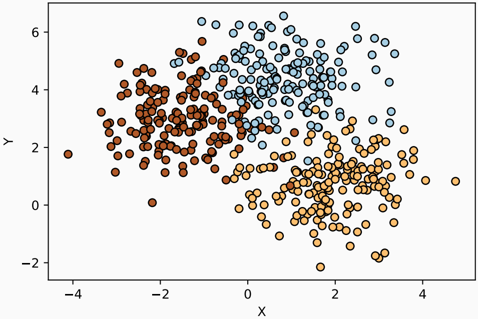
With an increase in the number of dimensions:

1. the distance between points starts to increase
2. the variation in distances between points starts to decrease

Due to this, distance metrics lose their efficacy as a similarity metric.

Approximate Nearest Neighbors

Doing an exhaustive search over all the points is time consuming, especially if you have a large number of data points.



Example of a big dataset

Approximate Nearest Neighbors

Doing an exhaustive search over all the points is time consuming, especially if you have a large number of data points.

If you are willing to sacrifice accuracy there are algorithms that can give you improvements that go into orders of magnitude.

Approximate Nearest Neighbors

Doing an exhaustive search over all the points is time consuming, especially if you have a large number of data points.

If you are willing to sacrifice accuracy there are algorithms that can give you improvements that go into orders of magnitude.

Such techniques include:

Approximate Nearest Neighbors

Doing an exhaustive search over all the points is time consuming, especially if you have a large number of data points.

If you are willing to sacrifice accuracy there are algorithms that can give you improvements that go into orders of magnitude.

Such techniques include:

- Locality sensitive hashing

Approximate Nearest Neighbors

Doing an exhaustive search over all the points is time consuming, especially if you have a large number of data points.

If you are willing to sacrifice accuracy there are algorithms that can give you improvements that go into orders of magnitude.

Such techniques include:

- Locality sensitive hashing
- Vector approximation files

Approximate Nearest Neighbors

Doing an exhaustive search over all the points is time consuming, especially if you have a large number of data points.

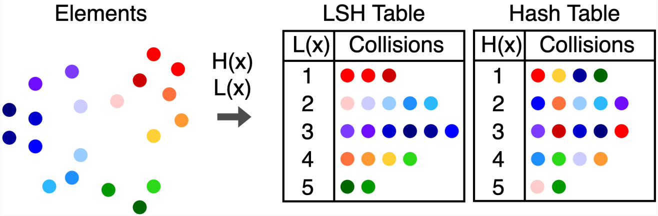
If you are willing to sacrifice accuracy there are algorithms that can give you improvements that go into orders of magnitude.

Such techniques include:

- Locality sensitive hashing
- Vector approximation files
- Greedy search in proximity neighborhood graphs

Locality sensitive hashing

Normal hash functions $H(x)$ try to keep the collision of points across bins uniform.

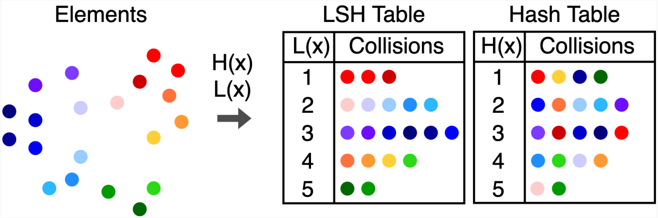


Example of a big dataset

Locality sensitive hashing

Normal hash functions $H(x)$ try to keep the collision of points across bins uniform.

A locality sensitive hash (LSH) function $L(x)$ would be designed such that similar values are mapped to similar bins.

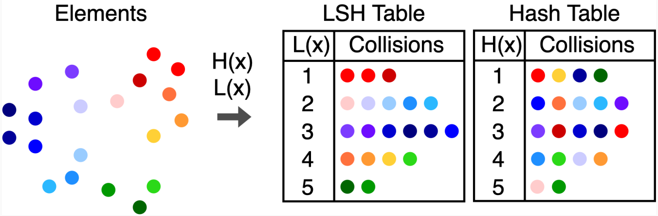


Example of a big dataset

Locality sensitive hashing

A locality sensitive hash (LSH) function $L(x)$ would be designed such that similar values are mapped to similar bins.

For such cases, all elements in a bin would be given the same label, which again can be decided on the basis of different aggregation methods



Example of a big dataset

null